计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(11):238−242 [doi: 10.15888/j.cnki.csa.005470]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于半边结构和 √3 细分的渐进网格生成方法①

马建平[1], 柴　毅[1], 陈　渤[2], 徐家园[1], 陈　强[3]

[1](浙江工业大学 计算机科学与技术学院, 杭州 310023)
[2](浙江商业职业技术学院 应用工程学院, 杭州 310053)
[3](广东第二师范学院 计算机科学系, 广州 510303)

摘　要: 渐进网格可以满足生成多分辨率模型的需求. 在现有渐进网格生成方法中, 一个顶点的简化往往关联四个以上的相邻顶点. 并且, 现有方法多采用网格的点面列表结构表示. 本文采用 √3 细分预测方法生成渐进网格, 每个顶点的存储仅关联三个相邻顶点. 同时也使用半边数据结构替代网格的点面列表表示形式, 加快了邻接信息查询. 实验结果表明, 本方法提升渐进网格的空间效率, 缩短渐进网格的生成时间.

关键词: 半边结构; √3 细分; 渐进网格; 网格压缩

引用格式: 马建平,柴毅,陈渤,徐家园,陈强.基于半边结构和 √3 细分的渐进网格生成方法.计算机系统应用,2017,26(11):238–242. http://www.c-s-a.org.cn/1003-3254/5470.html

Progressive Mesh Generating Method Based on Half-Edge Structure and √3 Subdivision

MA Jian-Ping[1], CHAI Yi[1], CHEN Bo[2], XU Jia-Yuan[1], CHEN Qiang[3]

[1](Computer Science and Technology School, Zhejiang University of Technology, Hangzhou 310023, China)
[2](Application Engineering School, Zhejiang Vocational College of Commerce, Hangzhou 310053, China)
[3](Computer Science Department, Guangdong University of Education, Guangzhou 510303, China)

Abstract: Progressive meshes will meet the requirements of generating multi-resolutions meshes of a 3D model. Among the methods available, more than 4 adjacent vertices are associated to simplify a vertex. Moreover, meshes are represented by vertex-face list structure, which has bad experience in searching neighbor information. In this paper, √3 subdivision method is introduced to predict the vertex to be simplified, and only 3 adjacent vertices are considered. In the meantime, a half-edge map is constructed to replace vertex-face list so as to speed up the neighbor information search. Experimental results show that the method proposed in this paper improves both time efficiency and space efficiency of generating progressive meshes.

Key words: half-edge structure; √3 subdivision; progressive mesh; mesh compression

三维模型正被广泛应用于计算机图形领域, 例如计算机辅助设计, 三维游戏和移动应用. 这些模型由点、边、面的集合组成. 初始密网格由建模系统或3D 激光扫描系统生成, 常常含有数以万计的面和点, 其存储和传输的代价非常巨大[1]. 另外, 针对不同的渲染需求, 初始网格需要按照不同的分辨率生成多分辨率网格模型 LOD(levels of details), LOD 技术就一3D 模型对象定义了多种分辨率的网格, 根据模型对象与观察者之间的距离远近确定相应的分辨率网格粗细. 与此同时, 由于不必一次存储所有 LOD 层次, 该技术提升了渲染性能[2,3], 但 LOD 所有层次数据均要存储. 针对这些问题, 提出了渐进网格压缩方法. 该方法将初始网格转化为一系列的多分辨率表示, 并通过恢复一系列多分辨率表示渲染不同 LOD 层级.

Hoppe[4]最先引入渐进网格的概念. 该渐进网格通过一系列边折叠操作生成. 如图 1 所示[4], 一次边折叠

① 基金项目: 国家自然基金 (61640222, 61772140); 广东科技计划项目 (2017A010101021); 广州市科技计划项目 (201604010049)
收稿时间: 2016-02-24; 修改时间: 2016-03-22; 采用时间: 2016-04-05

操作将两个或两个以上顶点合成一个顶点. 图中 V_t 和 V_s 被并作了点 V', 于是, 初始密网格 M^n 能被简化为一系列粗网格 ($M^{n-1}, M^{n-2}, \ldots, M^0$, M^0 是最粗的网格).

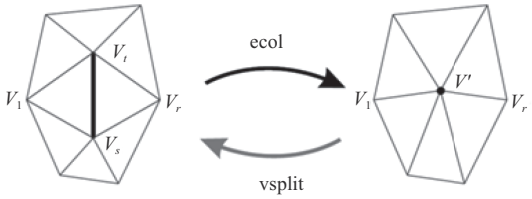

图 1 边分裂和边折叠变换

渐进网格不仅可以保持初始网格的几何特性, 而且保留了全部外观特征[4]. 根据边折叠结果, 任意三角网格网格 M^n 可以由基网格 M^0 和一系列顶点分裂操作重构. 例如, M^n 的渐进网格表示为 $M^0, \{Vsplit_0, \ldots, Vsplit_n\}$.

Hoppe 的渐进网格模型提供了渐进网格表示方法的基本原则. 在该基本原则下, C.MazettoMendes 等通过使用纹理映射提出了数据驱动的渐进网格生成算法[5], 但该算法需要求解复杂方程, 运算量较大. 与此同时, 研究者提出了多种预测策略来解决不平衡问题并提升简化性能. 例如线性最小均方误差 (MMSE) 策略[6], 逆 butterfly 细分策略[7], 逆改型 LOOP 细分策略[8]等. 这些算法中, 边折叠操作通过预测策略进行顶点删除. 预测被用于计算将删除顶点的坐标位置, 并保留预测坐标位置和原坐标位置的误差. 但是, 这些算法进行预测时都关联了 4 个以上相邻顶点. 此外, 由于预测过程中保存误差时尚须记录邻接顶点的信息, 顶点数量会大大影响渐进网格的数据量.

另一方面, 三维网格一般由顶点列表和面列表表示. 当简化操作应用于某特定顶点时, 面列表和顶点列表须要被遍历多次来查找该顶点的邻接顶点. 为解决时效问题, 在渐进网格生成过程中, 本文使用半边数据结构[9]替代传统结构. 半边数据结构只须常数时间查找某顶点的邻接信息, 大大降低了渐进网格生成和重构的时间. 基于此, 本文提出了一种基于半边结构和$\sqrt{3}$细分的渐进网格生成方法, 实验数据表明, 本方法较诸其他传统方法, 具有更好的性能.

1 相关理论

顶点预测是解决不平衡问题和提升简化性能的有效方法. 常用的预测策略有逆改版 LOOP 细分预测策略和线性最小均方误差 (MMSE) 预测策略. 它们都通过若干保留顶点坐标计算冗余顶点的坐标位置, 而后删除冗余顶点, 重新网格化保留顶点得到简化网格.

逆改版 LOOP 细分预测策略, 如图 2(a).

计算预测点坐标的计算公式如下:

$$V_j^{LOOP} = \left(V_{j,1} + V_{j,3}\right)/8 + 3*\left(V_{j,2} + V_{j,4}\right)/8 \quad (1)$$

其中, $V_{j,1}$、$V_{j,3}$是与 V_j 直接相连的顶点, $V_{j,0}$、$V_{j,2}$不与 V_j 直接相连, 为 V_j 对边的邻面上的顶点.

MMSE 预测策略, 如图 2(b), 计算预测点坐标的计算公式如下:

$$V_j^{MMSE} = \sum_{k=0}^{7} \omega_{j,k} V_{j,k} \quad (2)$$

其中, $\omega_{j,k}$是加权系数, 还需要通过求解复杂的矩阵方程[5], 计算量很大, 但是预测顶点位置较为精准.

逆$\sqrt{3}$细分预测策略, 如图 2(c), 计算预测点坐标的计算公式如下:

$$V_j^{\sqrt{3}} = \left(V_{j,0} + V_{j,1} + V_{j,2}\right)/3 \quad (3)$$

其中, $V_{j,0}$、$V_{j,1}$、$V_{j,2}$均与 V_j 直接相连.

逆改版 LOOP 细分预测策略需要考虑四个顶点进行预测, MMSE 预测策略需要考虑 8 个顶点, 且需要计算复杂矩阵方程计算加权系数来进行预测. 本文提出的逆$\sqrt{3}$细分预测策略仅需考虑三个顶点进行预测, 因此其效率更高.

此外, $\sqrt{3}$ 细分是已知细分策略中细化速率 (refinement rate) 最慢的, 避免了渐进网格重建时面片数骤增的情形. 同时, 它保持局部一致性, 极限情况下可以生成具有 C2 阶光滑的曲面, 这使得渐进网格重建时保持光顺.

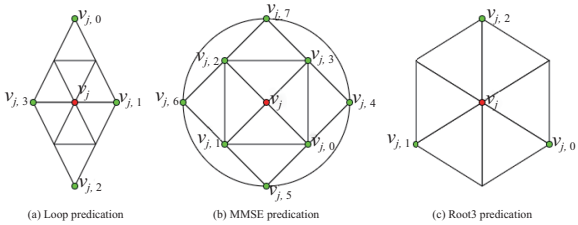

图 2 预测模板

2 各模块的算法设计与实现

2.1 半边映射构建

较诸点面列表表示的三维网格存储结构, 半边结构需要存储点、面、边的邻接信息, 因此更为复杂. 使用半边结构可以令邻接信息查询在常数时间完成[10].

在半边结构中, 网格的边用两条方向相反的半边表示. 边的顶点序号顺序表示边的方向.

本文的半边数据结构用哈希映射方式实现. 就某一半边而言, 该半边的顶点索引序号作为映射键, 该半边的起始点坐标、左三角面、逆边、下边作为映射值. 半边映射的构建伪代码如图 3 中所述.

```
buildFaceList(fileName, faceList);
Face face0 = faceList[0];
// 任意顺序构建初始面的三条半边
HalfEdge e01,e02,e03 <= face0.buildEdges();
// 将该三条边加入半边映射
HE_Map.putEdges(e01,e02,e03);
HE_Queue.enqueue(e01,e02,e03);
while(!HE_Queue.empty())
  HalfEdge e = HE_Queue.dequeue();
    if (!HE_Map.get(e.oppsite()))
      Face face = e.getAdjacentFace(faceList);
      // 按照 e 的反方向构建 face 的三条边
      HalfEdge e1,e2,e3 =
        face.buildEdgesWithDirection(e.opst);
      HE_Map.putEdges(e1,e2,e3);
      HE_Queue.enqueue(e1,e2,e3);
```

图 3 半边映射构建伪代码表述

就网格 M(V_{num}, F_{num}) 而言, V_{num} 表示该网格的顶点数, F_{num} 表示该网格的面片数. 该构建过程中, 由于每一条半边都要搜索其对应的逆边, 因此构建算法的时间复杂度为 O(F_{num}^2), 空间复杂度为 O(V_{num} + F_{num}).

2.2 逆√3细分生成渐进网格

文献[11]提出了渐进网格的生成策略. 对于给定密网格, 该策略通过顶点划分、奇点预测和重新网格化三步骤逐层生成渐进网格

2.2.1 顶点划分

文献[11]将三角网格中度为 6 的顶点定义为正则点, 度不为 6 的顶点定义为奇异点. 在顶点划分过程中, 所有奇异点都被划为偶点. 此外, 与偶点直接相连的顶点都为奇点. 顶点划分时, 若有奇异点则任选一个奇异点作为起始点, 否则任选一个顶点作为起始点, 如图 4 中的 V_e, 将其设为偶点. 接着将与 V_e 直接相连的顶点设为奇点. 然后使用半边映射, 根据半边<V_{02}, V_{01}>找到 V_e', 将其设为偶点. 如此往复, 直到所有顶点都被标记, 划分为偶点和奇点两个顶点集合.

2.2.2 奇点预测

顶点划分将顶点划分为偶点和奇点. 所有奇异点都在偶点集中, 奇点均为正则点, 可以根据公式 3 逆/3 细分预测器进行预测奇点位置$V_j' = \left(V_{j,0} + V_{j,1} + V_{j,2}\right)/3$, 并计算实际奇点与预测奇点之间的偏差, 得到偏移量:

$$d_j = V_j - V_j' \tag{4}$$

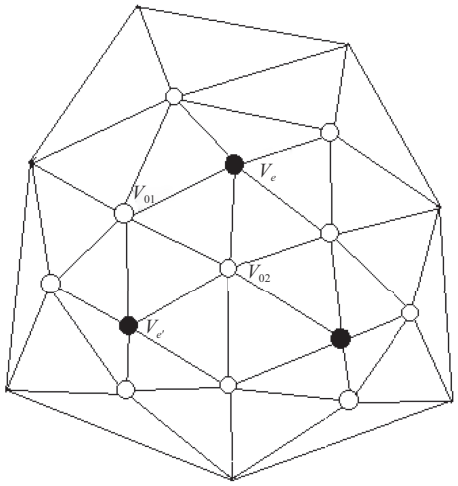

图 4 顶点划分示意

偏移量 d_j 用当前奇点的索引序号 V_j、三个关联顶点的索引序号$V_{j,0}, V_{j,1}, V_{j,2}$以及偏移值表示.

2.2.3 重新网格化

奇点用偶点进行关联后被删除. 对偶点进行重新网格化得到简化网格.

如图 5 所示, 网格 M^j 中的虚线边将被移除, 实线边构成新的简化网格 M^{j-1}. 图 5 中的 12 个三角面片简化为 4 个面片. 由于 M^j 网格中的奇异点都被保留下来, 因此简化后的粗网格 M^{j-1} 基本保持上一层网格的特征.

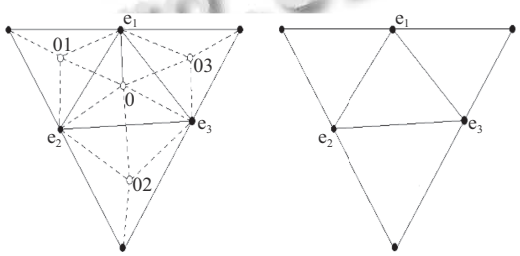

图 5 重新网格化

重复上述 3 个步骤, 可以将初始密网格 M^n 逐步简化为基网格 M^0, 即生成了 LOD 网格模型 $M^n \to M^{n-1} \to \ldots \to M^1 \to M^0$, 同时还生成了一系列的偏移量 d, 渐进网格表示为$M^0, d^0, d^1, d^2, \ldots, d^{n-1}$.

2.3 网格重建

网格的重建过程是网格简化的逆过程. 如图 6 所示, 首先, 根据基网格 M^0 构建网格的轮廓模型, 再用√3细分策略进行网格细分, 细分过程中产生的新顶点用偏移量 d^0 进行校准, 得到网格 M^1. 重复以上步骤, 直

到完全重建 M^n, 或者所得模型 $M^i(0<=i<n)$ 的分辨率满足用户需求为止.

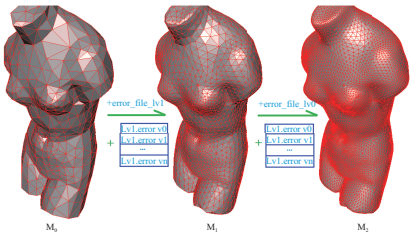

图 6 渐进网格重建

3 实验

本方法生成渐进网格的过程如图 7 所示.

图 7 预测模板

本文用渐进网格生成时间和压缩比率两个维度来测试算法的性能. 在实验中, 将该算法与两个最近提出的渐进网格生成算法, 即逆 LOOP 细分算法和 MMSE 算法进行比较. 实验环境为 PC, 其 CPU 为 Intel Core i5-2410@2.30 GHz, 4 G 内存.

表 1 三方法渐进网格生成时间效率比较

Mesh	Face Num	本文算法	LOOP	MMSE
		Time (s)	Time (s)	Time (s)
Head	22688	1.75	6.50	11.95
Venus	9504	0.52	1.24	2.26
Cow	15904	1.09	3.19	6.22
Bunny	6336	0.55	0.59	1.05
Cessna	21856	1.52	6.03	11.22
Gourd	10368	0.57	1.42	2.59
Shuttle	9792	0.58	1.32	2.30

表 1 比较了三个算法渐进网格生成时间的长短. 显然, 本算法生成测试模型的渐进网格时间最短. 此外, 尽管测试模型的三角面片数相差巨大, 本算法的性能受到的干扰很小, 而另外两个算法波动甚巨. 这是由于在顶点预测过程中, 半边映射较诸点面列表表示, 省去了不必要的顶点或面片遍历查询. 使用半边映射, 可以使得这些查询都在常数时间完成. 当然, 半边映射需要额外的存储空间, 但是由于渐进网格生成的过程是一次性的, 不必每次使用渐进网格都生成一次, 因此这些空间开销可以忽略.

表 2 三方法渐进网格生成空间效率比较

Mesh	Face Num	本文算法	LOOP	MMSE
		Ratio (%)	Ratio (%)	Ratio (%)
Head	22688	19.1	25.9	22.9
Venus	9504	24.1	41.8	25.7
Cow	15904	24.0	35.0	24.6
Bunny	6336	41.0	56.2	43.5
Cessna	21856	21.1	36.1	25.0
Gourd	10368	19.3	40.0	22.4
Shuttle	9792	23.7	36.7	29.0

就压缩比率而言, 本算法明显优于逆 LOOP 算法, 略优于 MMSE 算法, 如表 2 所示. 这是由于 MMSE 算法在预测奇点坐标时, 考虑了更多的相邻顶点 (8 个顶点) 来预测奇点位置, 因此 MMSE 算法预测所得的奇点坐标更为精准, 其保存的偏移量就会比较小. 但是由于考虑顶点众多, 该算法进行网格重建时计算量大大增大.

4 结语

本文提出了一种基于$\sqrt{3}$细分策略和半边数据结构的渐进网格生成方法. $\sqrt{3}$细分策略提供了奇点预测策略, 基于半边结构模型构造的半边映射解决了渐进网格生成过程中邻接信息查找的时效问题. 实验结果表明, 本方法可以缩短渐进网格生成时间, 减少存储空间. 在未来的工作中, 将结合渐进网格编码算法, 例如嵌入式小波零数编码[11], 来提升压缩效果.

参考文献

1 Aviles M, Moran F. Static 3D triangle mesh compression overview. Proc. of the 15th IEEE International Conference Image Processing, 2008. San Diego, CA, USA. 2008. 2684–2687.

2 Lee J, Choe S, Lee S. Compression of 3D mesh geometry and vertex attributes for mobile graphics. Journal of Computing Science and Engineering, 2010, 4(3): 207–224. [doi: 10.5626/JCSE.2010.4.3.207]

3 Berjón D, Morán F, Manjunatha S. Objective and subjective evaluation of static 3D mesh compression. Signal Processing: Image Communication, 2013, 28(2): 181–195. [doi: 10.1016/j.image.2012.10.013]

4 Hoppe H. Progressive meshes. Proc. of the 23rd Annual Conference on Computer Graphics and Interactive Techniques. New York, USA. 1996. 99–108.

5 Mendes CM, Apaza-Agüero K, Silva L, *et al*. Data-driven progressive compression of colored 3D mesh. Proc. of 2015 IEEE International Conference on Image Processing. Quebec

City, QC, Canada. 2015. 2490–2494.
6 Lee DY, Ahn JK, Ahn M, *et al*. 3D mesh compression based on dual-ring prediction and MMSE prediction. Proc. 2011 the 18th IEEE International Conference on Image Processing. Brussels, Belgium. 2011. 905–908.
7 Kammoun A, Payan F, Antonini M. Sparsity-based optimization of two lifting-based wavelet transforms for semi-regular mesh compression. Computers & Graphics, 2012, 36(4): 272–282.
8 马建平, 罗笑南, 陈渤, 等. 一种面向移动 3D 图形的几何简化方法. 计算机研究与发展, 2008, 45(8): 1395–1401.
9 Campagna S, Kobbelt L, Seidel HP. Directed edges—a scalable representation for triangle meshes. Journal of Graphics Tools, 1998, 3(4): 1–11. [doi: 10.1080/10867651.1998.10487494]
10 McGuire M. The half-edge data structure. http://www.flipcode.com/archives/The_Half-Edge_Data_Structure.shtml. [2008-08-07].
11 马建平, 罗笑南, 陈渤, 等. 面向移动终端的三角网格逆细分压缩算法. 软件学报, 2009, 20(9): 2607–2615.